ONBOARD AUTONOMOUS NAVIGATION ARCHITECTURE FOR A PLANETARY SURFACE EXPLORATION ROVER AND FUNCTIONAL VALIDATION USING OPEN-SOURCE TOOLS

Raul Correal [(1)], Gonzalo Pajares [(2)]

[(1)] *TCP Systems and Engineering. Aerospace Systems Division. Fernandez Caro, 7. 28027 Madrid, Spain. Email: rcorreal@tcpsi.es*

[(2)] *University Complutense of Madrid. Dept. Software Engineering and Artificial Intelligence. C/ Profesor José García Santesmases, s/n. 28040 Madrid, Spain. Email: pajares@fdi.ucm.es*

ABSTRACT

In the planetary domain there exist severe communication constraints linking planets like Mars and the ground station on Earth, such as signals delays and communication windows. This makes critical the necessity of having onboard local autonomy, allowing deployed vehicles making its own decisions independently of ground control, maximizing the scientific return of the mission and reducing operational costs and risks. This paper outlines a visual-based autonomous navigation approach and software architecture for exploration rovers' onboard autonomy in planetary environments. A framework with simulation and monitoring capabilities is presented, developed to support this research, allowing analysis of performance and behaviors to evaluate feasibility of strategies and early functional validation of approaches. Some of these capabilities are partially supported by COTS and open-source packages.

1. INTRODUCTION

The concept of autonomy is commonly understood as the capacity of a system to make decisions on its own. In the planetary domain, once a robotic vehicle has been deployed on the targeted planet surface, it must interact with many non-predictable local conditions. Autonomy entails the capacity of dealing with those situations without any or minimum dependency on ground control.

The constraints on communications, with signal' delays up to 40 min. from Earth to Mars, and the limited availability of satellites orbiting the target planet makes teleoperation unfeasible and imperative the necessity for onboard autonomy. Among the multiple operations a planetary rover shall perform, navigation is one of the most critical, when the vehicle must drive from one location to another performing on-site scientific investigations and is exposed to risks such as tipping over, falling in cracks or getting stuck in soft sand, putting the whole mission at risk. For that reason, rovers are typically operated in a very conservative way, elaborating plans from ground control to be daily uploaded to the rover, allowing a restricted amount of onboard autonomy and just under certain conditions.

That is the case of the NASA MER mission [1], where two twin rovers, Spirit and Opportunity, were deployed on opposite sides of Mars in January 3 and January 24, 2004, with the primary goal of searching for and characterizing a wide range of rocks and soils that hold clues to past water activity on Mars. The mission, originally scheduled for 3 months duration, resulted in an unprecedented success, being at the time of this writing operating for more than 7 years.

The European Space Agency (ESA) will launch its first Mars robotic exploration mission, Exomars, by 2018, deploying two different rovers at the same location, Exomars and MAX-C, carrying complimentary scientific instruments. The main focus of the ExoMars Rover is to search for evidence of past and present life on Mars, drilling the soil to collect samples and analyzing molecules to study the Martian geology and mineralogy, and search for biosignatures. This rover is expected to be highly autonomous. Scientists on Earth will designate target destinations on the basis of compressed stereo images acquired by the cameras mounted on the rover mast. It must then calculate navigation solutions and safely travel approximately 100 m per sol. To achieve this, it shall create digital maps from navigation stereo cameras and compute a suitable trajectory avoiding collisions and hazards. The onboard intelligence for this explorer vehicle is currently under development [2]. Some activities related to rover control, such as CNES' EDRES autonomous navigation framework evaluation as potential flight software candidate [3] have already been done.

This paper presents the current state of the works we have carried out in the development of a planetary exploration rover's control architecture. In section 2, the visual-based autonomous navigation internal processes are described, so the robot plans its actions and makes its own decision by perceiving its surroundings, computing trajectories and executing them safely. Section 3, presents an overview of the onboard software architecture and how processes are organized. In section 4, simulation facilities developed for testing and validation are outlined.

2. VISUAL-BASED AUTONOMOUS NAVIGATION

As denoted before, navigation is one of the rover's most critical activities, where the whole mission is put at risk. When working autonomously, the security and integrity of the vehicle directly depends on which areas and how the rover decides to traverse. Safe paths must be computed, avoiding obstacles and hazardous situations such as tipping over, getting stuck or losing traction. Terrain assessment and computation of paths directly depends on the environment's internal representation, which is a direct result of the perception process. In planetary environments, robotic explorers commonly rely on the use of cameras to sense the environment and build a digital representation.

2.1. Sensing the Environment

To obtain 3D information from the environment, a pair of cameras, or a stereo camera, is commonly used, taking two images (left and right) at the same time. To match pixels in both images and compute disparities, we use a block matching algorithm [4]. It works with previously rectified images, using small "sum of absolute difference" (SAD) windows, finding strongly matching (high-texture) points, maximizing the number of pixels used subsequently to compute depth, see fig. 1.

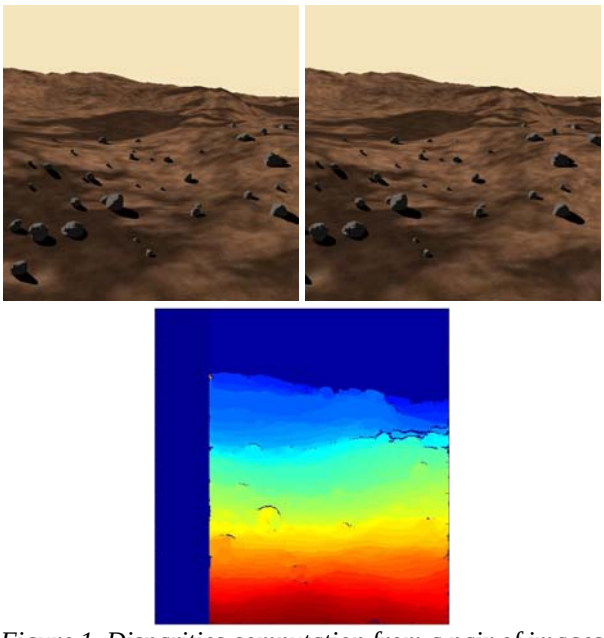


Figure 1. Disparities computation from a pair of images

A previous process normalizes brightness in images, reducing lighting differences and enhancing texture based on neighbors' value on a given window, maximizing chances of finding right correspondences.

After correspondences computation, a filtering phase eliminates false matches, overcoming random noise and avoiding speckle, local regions of large and small disparities. An additional clustering filter eliminates the still remaining bad matches, detecting isolated clusters of matches with no spatial continuity. Fig. 2 shows the effect of this filter, appreciated in the left-bottom corner.

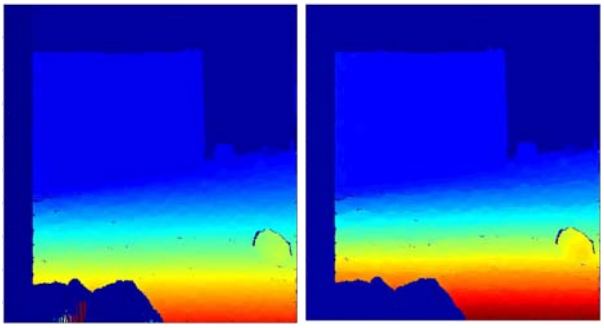


Figure 2. Disparity filtering using clusters

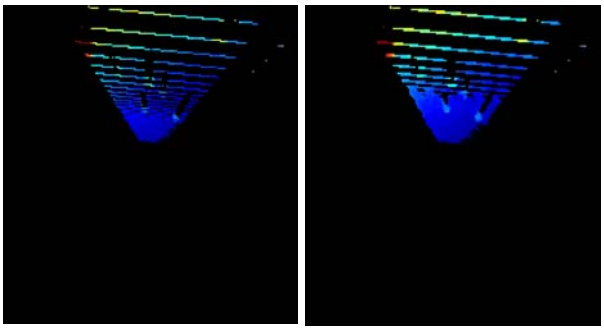
Knowing the geometric arrangement of the visual system, depth data is derived from the filtered disparities by triangulation, where each pixel is translated into depth, getting Z values. The result is a 3D points cloud representing the environment from the camera point of view. A series of geometric transformations translate these points to the rover reference system for 3D terrain reconstruction.

When working in simulation with synthetic images, as it is further described below, the rectification process is not necessary given they don't have distortions caused by camera lenses. Also the calibration process to obtain the intrinsic parameters of the camera like focal length, principal point or distortion models is not necessary given they are known a priori. However, these processes have been implemented for later use with real images.

The onboard software architecture has been designed to be parametric, easily controllable by means of a configuration file, establishing parameters and values so many different configurations can be quickly set in order to test and evaluate its impact and performance on the overall navigation process. Regarding the perception process, some of configurable parameters are camera resolution, focal length, field of view, offset between perceptions, correspondence algorithm in use, etc.

2.2. Environment Representation

The 3D points cloud resulting from the stereo process is not an appropriate method of representing the environment for terrain assessment and trajectory computation. A most appropriate technique is building digital elevation models (DEMs). It encapsulates elevation data on a grid structure, initially empty, filled with the 3D points. The result is a rover-centered top-view representation of the environment, as showed in fig. 3(a).



(*a*) (*b*)

Figure 3. a) Initial DEM; b) interpolated DEM

There is usually no enough information to fill up the whole grid, with some areas remaining unknown. These areas will have an impact in later trajectory planning. To minimize its impact, some empty portions of the map can be estimated based on its surroundings. This process can be seen in fig. 3(b), showing the resulting DEM after interpolation. The method to be used and parameters, such as number of neighbors to consider, are configurable.

Initially the DEM is empty and, as the rover performs more perceptions, new portions of the environment are discovered and the internal map is extended and updated. Except at the very beginning, where three pairs of images are taken to initialize the map -front and both sides of the rover, two new perceptions are done at each navigation cycle.

In order to cover the widest possible area, a configurable offset between perceptions is established, usually with some overlapping to avoid unknown areas. For instance, if the camera is modeled with a 90º HFOV (Horizontal Field of View), configuring a 35º offset allows imaging almost the whole front terrain, fig. 4.

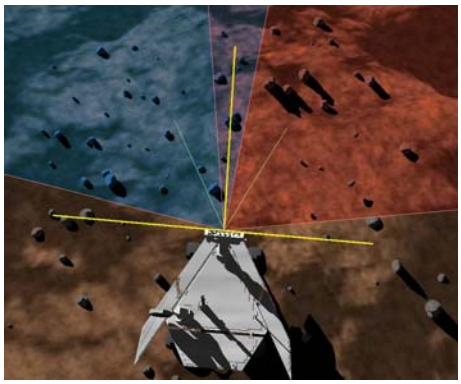
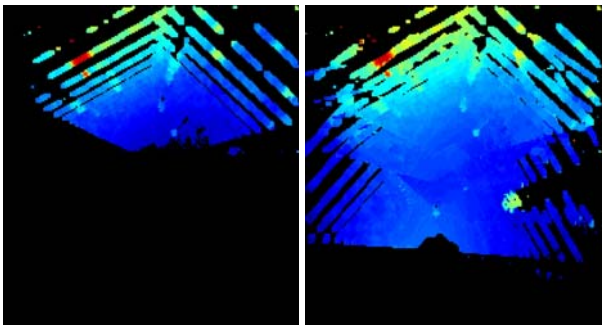


Figure 4. Terrain area captured by two perceptions (red and blue) with an overlapping region (purple). Green lines represent direction of perceptions

New data has priority over old data. Previous map is updated to keep the rover at the DEM center, and merged with new information, taking into account rover and pan/tilt unit attitude when images were captured. Fig. 5 shows a terrain model built from two single merged perceptions and the integrated map after several navigation cycles.

When merging data from two different perceptions from the same location it may happen more than one 3D point computed are related to the same grid cell. Each point may have different elevation data. In such cases, different strategies may be implemented like computing elevation average, taking the highest of lowest value or the largest magnitude -elevation or depression. The later is the one followed in these examples.



(*a*) (*b*)

Figure 5. a) Merged DEM from two and b) after several perceptions

Only local distance around the robot is stored in the map, forgetting old and distant data. The above examples represent a 10*x*10 meters terrain with a resolution, cells size, of 5 cm. Values such as map size, resolution, interpolation method and its parameters can be configured by the operator.

2.3. Trajectory Computation

The ultimate goal of perceiving the environment and building an internal representation is to compute paths so the rover drives safely from one location to another. Some points must be taken into account in this problem domain when designing a suitable trajectory computation algorithm: 1) the map is not complete, there exists unknown areas not seen yet by the robot, 2) the target location may be beyond map limits -the robot should go 12 meters ahead and the internal map stores 10 meters, 3) the world is not binary, meaning each cell doesn't have just two possible values, free or occupied, but a range value representing elevation data, 4) the robot is not a point, several cells are involved when computing rover safety at a given location, 5) the rover is non-holonomic. It implies some conventional path planning methods and search algorithms cannot be applied under these constraints.

Our approach is based on the GESTALT local planner [5] (Grid-based Estimation of Surface Traversability Applied to Local Terrain), developed by NASA-JPL for

the MER' mission rovers, Spirit and Opportunity. The algorithm projects a series of candidate paths on the grid and evaluates its suitability. Paths are equidistantly separated. The list of cells traversed by each one is obtained and the traversabilty of each cell evaluated.

For each of those cells, rover integrity must be evaluated in order to declare the whole path safe. In fig. 6, red cell indicates the rover center. Green cells delimit the area under the wheels, delimiting the portion of terrain interacting with the vehicle. For simplicity and conservatism, the algorithm shall account for all possible rover' poses centered at that location. To do that, rover-size diameter circumference is computed, represented by the yellow circumference, which includes the set of cells underneath the rover, being this at any possible pose and therefore, the area to be evaluated to assess rover safety.

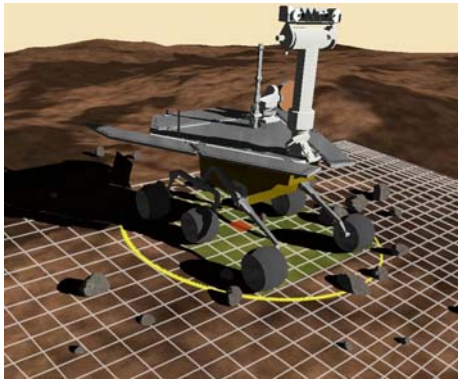


Figure 6. DEM area affected computing rover safety

The next values are computed, taking into account the whole set of cells within the affected area, and assigned to the central cell:

- Step: maximum elevation difference between any pair of adjacent cells within the circumscribing area.
- Roughness: maximum elevation difference among all cells within the circumscribing area.
- Tilt: maximum slope computed as a function of elevation difference among peripheral cells within the circumscribing area, where front and back wheels rest.

Only cells part of candidate paths are evaluated, to avoid spending computing power –and time- calculating cells' values, the whole map, which will never be used. Once these values have been obtained for every cell of every path, the direction deviation of each route with respect to the goal is computed. The algorithm tries to find the more directed path to the goal meeting the established security criteria. First, the path with minimum absolute deviation is chosen. If security constraints are met the path is selected. Otherwise, the process is repeated with the next one with minimum absolute deviation.

When a path is selected, the algorithm computes the relative deviation between it and the current robot heading to be sent to the low level controller, along with path data, to execute the trajectory. In case no path is selected, several strategies can be followed, as described in section 2.4.

Security constraint values -step, tilt and roughness hazards, are established by the operator. Configurable parameters for this module also are: number of forward and backward candidate paths, type (arcs, lines, etc), planning distance, navigation distance, chassis measures, body clearance or wheels' separation.

Planning distance refers to the candidate paths' length and navigation distance specifies how long the selected path will be actually followed. The planning distance is commonly set larger than the navigation one. The reason behind this is to avoid getting too close to obstacles and non navigable areas that may subsequently cause the rover moving backwards, see fig. 7. Outer area represents planning distance and inner the navigation area.

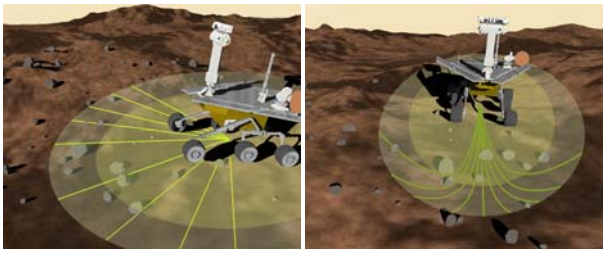


Figure 7. Path planning process, with straight and arc candidates

Chassis measures has a direct impact in establishing which cells are underneath the rover at any given position and have to be evaluated in the path planning process. Body clearance determines what areas can be traversed. As these parameters are configurable by the operator, the impact they have on the whole navigation process can be evaluated straightforward.

2.4. Navigation

The navigation process is initiated when a daily plan, containing scheduled actions, is received from ground control. By now, as our research is focused on autonomous navigation capabilities this plan consists just on a series of waypoints the rover has to go. For each waypoint, the rover determines the perception direction. In case it is out of sight, meaning out of the area that can be perceived by two frontal overlapped perceptions, see fig. 4, the mast is rotated so the

waypoint is directly faced. Then the process described in previous sections begins, perceiving the environment, updating the digital elevation map, computing candidate paths and selecting the best one meeting all criteria.

It is interesting to point out the distance between waypoints received from ground control may not be the same as the rover configured planning/navigation distance. In such cases it's necessary to perform several navigation cycles to go from one waypoint to the next. In fig. 8, green areas represent the planning distance - usually shorter than navigation distance. In this example, two navigation cycles are required between waypoints, marked by blue flags.

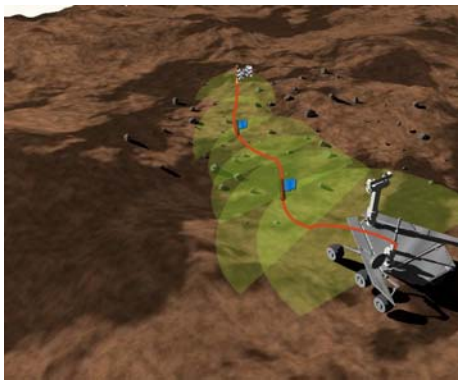


Figure 8. Several navigation cycles may be needed to go from one waypoint to the next

If case no suitable path is found among the candidates, several strategies can be established by the operator: 1) report an error and wait for a new plan, 2) repeat the planning process increasing the number of candidate paths, 3) make new perceptions to extend and update the environment internal model, 4) compute backwards paths to move away from the non navigable area and plan again. More suitable strategies may be established.

In case of computing backwards paths, chances are the terrain has already been perceived and stored within the map. In such cases, paths can be computed without taking new images and the path can be followed moving back blindly. Otherwise, the camera, or the robot, is rotated to face the new area and take images to update the map.

The operator can configure locomotion parameters such as nominal and maximum linear/angular velocities and navigation parameters such as maximum steering angle or accuracy requirements determining a waypoint has been reached, established as an acceptable area.

3. ONBOARD SOFTWARE ARCHITECTURE

The most widely known robot control architecture is the layered design, or subsumption architecture, introduced by MIT Prof. Brooks [6]. This conventional design has been extensively used in many robotic developments, and even some adaptations of this model have been applied to planetary rovers' control, like the CLARAty architecture, developed by JPL [7], or LAAS [8].

The software architecture onboard our vehicle, designed to organize the processes described above, follows this same layered model. It's decomposed in levels containing independent modules with defined interfaces. Each module, or subsystem, encapsulates a concrete functionality. The main idea behind this modular design is to allow scalability and extensibility of the system, to be used as a research platform. A scheme of this architecture can be seen in fig. 9.

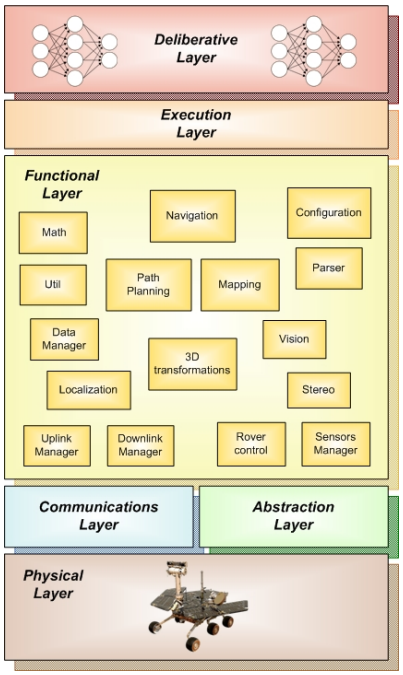


Figure 9. Onboard software layered architecture

The lowest layer is the physical level, where devices are electronically connected using the appropriate ports and buses. Communications and abstraction layer are placed over. The former provides means of contacting with ground station. This link is currently supported by ICE [9], an object-oriented toolkit that enables building distributed applications, being out of scope of this work modeling features such as satellite latency, delays or bandwidth constraints.

The abstraction layer encapsulates the access to underlying hardware and devices, in such a way it is transparent for the high-level software the access to the specific piece of hardware, by means of predefined interfaces. This allows easy future adaptations, in cases such as replacing devices from different manufacturers (e.g.: cameras) or adapting the autonomy architecture to different vehicles (e.g.: MSL or Exomars). The abstraction layer is provided by Player [10], which

provides a network interface to a variety of robot and sensor hardware, allowing writing control programs and supporting a wide variety of mobile robots and accessories.

Above it, the functional layer, which is the actual focus of the work presented on this document, is structured in individual modules. These components are organized and distributed by task, embedding the whole rover functionality described in previous sections.

These modules have been designed to have a high level of cohesion and low interdependency, where interactions among them are supported by defined interfaces. For instance, the module creating a digital elevation map from a 3D points cloud is independent from the module that generates the cloud, in this case the stereo vision process, so in the future the insides of the stereo process may be easily modified, extended or substituted, or a new one using a laser scanner to produce 3D points can be added, as long as it fulfills with the interface.

A light plan execution mechanism coordinates the execution of actions. Currently, a sequential control flow instantiates the functions of the corresponding modules in a sense-plan-act paradigm, perceiving the environment, updating its internal representation, planning trajectories and executing them. On top, a high-level deliberative layer controls overall rover activities. The role of this decisional level is to create global actions plans considering mission constraints, time and available resources, monitoring operations, checking for plan deviation, dealing with unexpected problems, evaluating risks and generating contingency actions to adapt the plan whenever necessary.

As our research has been focused so far on the functional layer, a sophisticated executive procedure or onboard planning haven't been developed at the time of this writing, currently generating navigation plans on ground station to be sent to the rover.

4. SIMULATION

A crucial point in any software development is testing. In order to validate the performance of the functional layer modules a set of experiments must be designed. Two main methods can be followed: either getting hold of a rover-like vehicle and a mars-like outdoor terrain or create a computer simulation.

Despite a system like this has to certainly be validated on the field before launching, at early stages, where strategies and approaches are initially designed, simulation is the most appropriate technique, testing with the real, and usually unique and expensive, system only when approaches are mature enough. Besides, the real vehicle may not be available till advanced phases of the development or there may not be enough resources to get a vehicle. Simulation has also the main advantages of repeatability and controllability, determining the concrete settings of the experiments as desired, including managing any aspect of the vehicle, terrain or conditions, reproducing the exactly same scenario as many times as necessary. These capabilities are crucial in validating the system

A key aspect in testing the system is closing the control loop; meaning modeling the plant to obtain the necessary data to feed the controller and sending the produced output back to the plant to simulate the effects it produces. In this case, the plant is a simulation of the rover, terrain and its interactions, and the controller is the rover onboard software. According to fig. 9, the model of the plant replaces the physical layer. The abstraction layer provides transparency to higher levels, which remain mostly unaltered, except for the extended functionality and necessary adaptations arising when moving from the simulated to the real world.

Within the control loop, sensor readings are obtained from the plant –stereo camera images, IMU data, etc, processed onboard and computed trajectories sent, decomposed in lower level motor commands, to the rover motors, to generate motion. Several navigation cycles are performed perceiving the environment, processing images, calculating paths, driving the rover to a certain location interacting with the environment and perceiving again from the new site.

Space agencies have developed utilities and simulation frameworks with different levels of fidelity and sophistication in the course of the last years. Some of the most relevant ones are ROAMS [11] -Rover Analysis, Modeling and Simulation, from NASA/JPL, EDRES [12, 3] -Space Exploration Robotics Development Environment, developed by CNES - French National Centre for Space Studies, and 3DROV [13], developed by TRASYS Space for ESA. Unfortunately, these environments are proprietary and not publicly accessible to the research community.

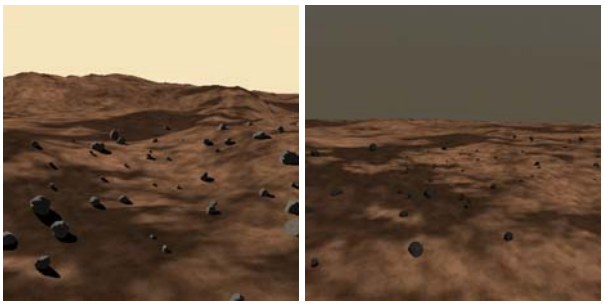
In this work, the system has been simulated using Gazebo [14], a multi-robot simulator with the capabilities of creating outdoor environments, robots with sensors and actuators and 3D real-time scene rendering. A set of sensor models are available, such stereo-cameras, and parameters like field of view, resolution and stereo base can be configured, so images of the environment from the camera point of view can be taken, to be used as input to the rover controller. A model of a planetary robotic explorer has been created based on the NASA MER rover, fig. 10. It has boogie suspension, six independent wheels, four of them

steerable, an IMU (Inertial Measurement Unit), a front low stereo camera and another one on top of a mast with two degrees of freedom –pan&tilt- for navigation.

Figure 10. Simulation model of the NASA-JPL MER rover

A Mars-like terrain, including mounds, depressions and rocks, has been created. The simulator allows controlling illumination conditions -light direction and intensity, creating shadows in the environment. A Dynamics engine computes rigid-body physics, modeling the rover-terrain interactions.

As stated along this document, one of the main design drivers on the system has been configurability. As it is addressed to be used as a research platform to study exploration autonomy strategies, an easy and quick configuration system is crucial. Besides the great deal of parameters mentioned along the text, the simulator allows setting values for terrain relief and complexity, soil texture, light direction and intensity, gravity, etc. and model related features such as rocks' size, pose and orientation, rover' wheels size, chassis measures, joints, torques, gains, lenses' field of view and resolution among others. Fig. 11(*a*) shows an image of a rough terrain taken from a 45º FOV stereo camera model and (*b*) an image of a softer terrain with a different rock distribution and illumination conditions taken from a 75º FOV stereo camera. The alteration of any of these parameters will have an impact on the internal rover's software computations and external behavior, being its analysis, characterization and measurement the main purpose of this research platform.



(*a*) (*b*)

Figure 11. Terrains, rock distribution, illumination conditions or camera models can be easily configured

Some measures on processes, algorithms performance and computing time have been obtained, tab. 1.

Table 1. Computing time on a PC Intel Core2 1.86 Ghz, with synthetic simulator-generated images (640x480)

Function	***Computing time***
Stereo matching	120-230 ms
Disparity filtering	40-90 ms
Computing 3D points	210-350 ms
Reprojection	100-220 ms
DEM construction	60-80 ms
DEM interpolation	< 10 ms
DEM updating	< 10 ms
Merge DEMs	< 10 ms
Path planning process	20-40 ms

The disparity filtering process is part of the stereo matching algorithm. Computing 3D points includes disparity matrix re-projection and perspective transformations to return a points list in the rover reference system. A complete perception process takes between 360 to 600 ms, including acquisition, stereo matching and 3D points' calculation. The computing time of a navigation cycle, the time the rover is idle with calculations, is 2-3 sec., excluding pan-tilt unit positioning, what is done three times on each cycle – left, right and back to front.

5. CONCLUSIONS

The main purpose of the presented work is developing a first approach of an autonomous navigation software architecture for a planetary exploration rover and the necessary infrastructure to support it, to be used as a research platform where more sophisticated and advanced functionality can be integrated over time to be tested and validated at the functional level.

The work has been focused on the functional layer, composed of independent modules communicated through defined interfaces, easing future modifications, extensions or, eventually, replacement. These modules are highly configurable, the operator can set many parameters and analyze its impact on the overall rover performance and behavior, which is extremely useful at research time when developing different approaches and strategies.

There are ongoing efforts to incorporate onboard planning, currently done on ground station, through the use of AI planning systems (i.e.: Fast-forward -FF), where aspects such as rover resources, power consumption, instruments usage, solar panels loading rate, sun position, contingent tasks, etc. are taken into account when designing autonomy strategies.

The Gazebo simulator has some limitations. However, it is important to emphasize the main purpose is to

analyze high-level autonomy strategies and validation of navigation approaches. It is not the aim of this research work studying aspects such as rover' mechanical design, terramechanics or advanced locomotion issues, which is not supported by the simulator, but, as denoted before, analysis of performance and validation of the functional layer modules, being the current models' level of fidelity appropriate to serve those purposes.

The ultimate validation has to be done by field testing. The architecture presented on this paper is currently being ported to flight-representative hardware. A Gaisler LEON III board, an outdoor sensorized mobile robot and stereocameras will be used to test the functionality of the modules and the whole autonomy process. Some developments and adaptations will take place to accommodate the current functionality to the new problems arising when dealing with the real world like illumination conditions, slipping soil, sensor noise, etc. Processes' performance will be measured and compared with the ones obtained from desktop simulation.

6. REFERENCES

1. Crisp, J. A., M. Adler, J. R. Matijevic, S. W. Squyres, R. E. Arvidson, and D. M. Kass. (2003). Mars Exploration Rover Mission. *Journal of Geophysical Research*, vol. 108, issue E12

2. Joudrier, L., Elfving, A. (2009). Challenges of the ExoMars Rover Control. American Institute of Aeronautics and Astronautics, AIAA 2009-1807. Seattle, Washington.

3. Odwyer, A., Correal, R. (2008). Experiences in Producing a Preliminary Navigation OBSW Prototype for the Exomars Rover Based on EDRES. In Proc. 10th ESA Workshop on Advanced Space Technologies for Robotics and Automation, European Space Agency, Noordwijk, The Netherlands.

4. Konolige, K. (1997) Small vision system: Hardware and implementation. In Proc. of the International Symposium on Robotics Research, Hayama, Japan, pp. 111–116.

5. Goldberg, S., Maimone, M., Matthies, L. (2002). Stereo Vision and Rover Navigation Software for Planetary Exploration. IEEE Aerospace Conference, Big Sky, Montana.

6. Brooks, R. (1986) A Robust Layered Control System for a Mobile Robot. *IEEE Journal on Robotics and Automation*, vol RA-2, no. 1.

7. Volpe, R., Nesnas, I.A.D., Estlin, T., Mutz, D., Petras, R., Das, H. (2001). The CLARAty Architecture for Robotic Autonomy. In Proc. Of IEEE Aerospace Conference, Big Sky Montana.

8. Ingrand, F., Lacroix, S., Lemai-Chenevier, S., Py, F. (2007). Decisional Autonomy of Planetary Rovers, *Journal of Field Robotics*, Volume 24, Issue 7, Pages 559 - 580

9. Henning, M., Spruiell, M. (2010). *Distributed Programming with Ice*. ZeroC, Inc.

10. Gerkey, B. P. Vaughan, R. T. Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proc. Int. Conf. on Advanced Robotics. Pages 317-323. Coimbra, Portugal.

11. Jain, A., Guineau, J., Lim, C., Lincoln, W., Pomerantz, M., Sohl, G., Steele, R. (2003). ROAMS: Planetary Surface Rover Simulation Environment. Int. Symp. on Artificial Intelligence, Robotics & Automation in Space. Nara, Japan.

12. Maurette, M., Rastel, L. (2002). Planetary rover simulation and operation. ASTRA, ESA Workshop on Advanced Space Technologies for Robotics and Automation. ESTEC, Noordwijk, The Netherlands.

13. Poulakis, P., Joudrier, L., Wailliez, S., Kapellos, K. (2008). 3DROV: A Planetary Rover System Design, Simulation and Verification Tool. 9th Int. Sym. on Artificial Intelligence, Robotics & Automation in Space. Los Angeles, USA

14. Koenig, N., Howard, A. (2004). Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. In Proc. of Int. Conf. on Intelligent Robots and Systems, Sendai, Japan.